QVT : un standard de transformation pour l'Ingénierie Dirigèe par les Modèles

Didier Vojtisek

QVT : un standard de transformation pour l'Ingénierie Dirigée par les Modèles

Introduction à l'ingénierie dirigée par les modèles et ses enjeux

Pour faire face à la complexité et à l'évolution croissante des applications, l'Ingénierie Dirigée par les Modèles (IDM) propose d'appréhender les applications selon différents points de vue. Pour pouvoir être appliquée, cette approche considère comme fondamental que les modèles représentant ces vues soient composés et mis en cohérences. Ils sont ainsi, placés au centre des activités de développement.

En effet, tous les domaines métiers peuvent être représentés sous forme de modèles. Une partie de l'activité du développement actuelle consiste déjà à traduire de l'information d'un modèle métier à un autre. Un exemple courant est le passage de la définition d'un système d'information vers sa représentation dans une base de donnée. Pour la représentation des modèles, l'Object Management Group (OMG) propose une série de métamodèles permettant de représenter la plupart d'entre eux. L'un des plus connu étant UML. Même dans les cas où les standards proposés ne permettent pas de représenter au mieux un domaine particulier, le MOF (Metadata Object Facility) permet toujours d'en définir un plus approprié.
Dans tous les cas, une part importante de l'activité consiste donc à transformer des modèles en d'autres modèles selon les besoins.
Ainsi, une partie du savoir-faire de l'entreprise est contenue dans ces transformations. Il est donc important de ne pas négliger ces transformations qui, correctement utilisées, permettent non seulement de pérenniser du savoir-faire mais aussi d'obtenir de bons rendements.

Présentation du futur standard QVT (Query/View/Transformation)

Suivant le processus de normalisation défini par l'OMG, les industriels, fournisseurs d'outils et les universitaires sont invités à soumettre des propositions. Pour QVT, les 8 soumissions initiales rassemblant plus d'une trentaine d'entreprises et institutions ont prouvé l'intérêt de ce standard. Après un long processus de convergence, le standard arrive actuellement en phase de finalisation et devrait être officialisé en juillet 2006.
Concrètement, et afin de satisfaire la grande majorité des nombreux participants, QVT a été construit de manière modulaire et combine plusieurs paradigmes.
Grâce à une architecture hybride, il propose ainsi les avantages combinés des approches déclaratives et des approches impératives.
De part ses capacités d'expression, la partie déclarative permet dans de nombreux cas courant d'écrire facilement des transformations entre modèles, en particulier lorsque la transformation est constituée d'un ensemble de règles relativement simples.
Néanmoins certains types de transformations se prêtent mal à un style déclaratif. C'est le cas par exemple lorsqu'une transformation nécessite de mettre en œuvre un algorithme utilisant des aspects combinatoires. Pour ce genre de cas, QVT propose alors un style impératif qui permet d'écrire des transformations dans un style plus traditionnel pour la majorité des programmeurs.

De plus, afin de profiter au mieux de la puissance d'expression des systèmes de règles, QVT défini deux niveaux déclaratifs :

- Le premier niveau est un noyau purement déclaratif qui permet, si on se restreint à cette partie, de gérer automatiquement des transformations bidirectionnelles.
- Le deuxième niveau est une partie relationnelle, qui apporte des mécanismes de plus haut niveau dont en particulier la possibilité de faire de la reconnaissance de motif dans un modèle. Il apporte aussi la possibilité de s'interfacer avec la partie impérative et une interconnexion plus facile avec l'existant.

QVT apporte en outre des mécanismes complémentaires tels que la génération de traces automatiques. Celle-ci aide à la résolution des problématiques récurrentes auxquelles les concepteurs sont confrontés lorsque les transformations s'appliquent en plusieurs passes successives.

Comparaison avec les pratiques actuelles, mise en lumière de certains choix technologiques et de leur signification

QVT répond au besoin de capitalisation du savoir faire actuellement éparpillé autour des différentes techniques de transformations. En effet, il est dès aujourd'hui possible de faire de la transformation de modèle.
On peut distinguer cinq grandes familles d'approches et utiliser :

- des langages génériques banalisés avec éventuellement le support de bibliothèques ou de framework dédiés. (ex : java, C#, …),
- des langages de transformation générique tels que ceux dédiés à la transformation de graphe ou XSLT…,
- des langages (souvent de script) associés à un CASEtool particulier tel que Objecteering ou Rose…,
- des outils de transformation dédiés tels que QVT lui-même,
- des outils de métamodélisation exécutable tels que Metacase, Kermeta ou Xactium…

Chaque pratique a ses propres avantages et inconvénients, mais la diversité des approches suivies rend la capitalisation difficile. L'une des forces majeures de QVT est d'une part de proposer un langage dédié à la problématique et d'autre part, de promouvoir une ouverture vers les autres approches grâce au support de transformations boîtes noires et à différents termes de compatibilité.

Les différents termes de compatibilité prévus par la spécification.

Le standard permet de définir les transformations comme des boites noires dont les spécifications sont écrites avec QVT alors que leurs implémentations peuvent être écrites avec QVT comme avec tout autre langage. Le système de boites noires répond aussi à un autre besoin important du marché : la diffusion de briques prêtes à l'emploi sous forme binaire. Il est en effet important de pouvoir échanger du comportement sans en montrer le contenu pour des raisons évidentes de confidentialité.
De plus, de nombreuses transformations existant déjà, il est coûteux pour les outilleurs d'adopter un standard qui les oblige à réécrire leur code existant. Le mécanisme de boite noire permet ainsi un certain type d'intégration puisqu'il suffit d'encapsuler et de définir le comportement extérieur de transformations déjà existantes.

De part la nature de QVT, Il existe plusieurs moyens d'être compatible. Ainsi la norme précise les critères qui seront utilisés pour indiquer si un outil ou une transformation est compatible avec la norme ou pas.
Comme QVT est défini dans une vision modèle, il utilise un format échange sous forme de modèle ou bien sous forme de grammaire textuelle. Le format d'échange sous forme de modèle est spécifié dans un autre standard de l'OMG : XMI. Un outil pourra ainsi exécuter ou exporter une transformation écrite avec la syntaxe textuelle de QVT ou en XMI.
Enfin, la structuration interne permet d'identifier des sous-ensembles exécutables que les outilleurs souhaiteront supporter ou pas. Ainsi on pourra trouver des outils supportant une ou plusieurs des dimensions du langage suivantes : le noyau déclaratif, la partie relationnelle et la partie impérative.

Ces mécanismes devraient permettre de passer progressivement vers ce nouveau standard.

Didier Vojtisek
Ingénieur de Recherche à l'INRIA
didier.vojtisek@irisa.fr
http://www.irisa.fr/triskell